AD-A101 042 MICHIGAN UNIV ANN ARBOR DEPT OF ELECTRICAL AND COMPU--ETC F/G 9/2
VECTORIZED SPARSE ELIMINATION.(U)
JUN 80 D A CALAHAN AFOSR-80-0158
UNCLASSIFIED AFOSR-TR-81-0542 NL

END
DATE FILMED
7-81
DTIC

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| 1 REPORT NUMBER: 19 AFOSR-TR- 81-0542 | 2 GOVT ACCESSION NO: AD-A101042 |
| | 3 RECIPIENT'S CATALOG NUMBER: 041 |
| 4 TITLE (and Subtitle): VECTORIZED SPARSE ELIMINATION. (1) | 5 TYPE OF REPORT & PERIOD COVERED: 9 Interim rept. |
| | 6 PERFORMING ORG. REPORT NUMBER |
| 7 AUTHOR(s): 10 D. A. Calahan | 8 CONTRACT OR GRANT NUMBER(s): 15 AFOSR-80-0158 |
| 9 PERFORMING ORGANIZATION NAME AND ADDRESS: University of Michigan, Dept. of Elec. & Comp. Engineering, Ann Arbor, MI 48109 | 10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 9749-03, 61102F 16 17 03 |
| 11 CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research, Bolling AFB, Washington, DC 20332 | 12 REPORT DATE: June 8, 1980 |
| | 13 NUMBER OF PAGES: 12 12 134 |
| 14 MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15 SECURITY CLASS. (of this report): unclassified |
| | 15a DECLASSIFICATION DOWNGRADING SCHEDULE |

LEVEL II

16 DISTRIBUTION STATEMENT (of this Report)
Approved for public release, distribution unlimited.

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)
Approved for public release; distribution unlimited

DTIC ELECTE JUL 7 1981 S C D

18 SUPPLEMENTARY NOTES

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)
Sparse matrices
Vector processors
Parallel processors

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report summarizes research investigating (1) algorithms for the vectorized direct solution of sparse simultaneous equations, and (2) the application of vector processors to electronic circuit analysis. The research was performed between 4-1-80 and 3-31-81.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified 407400

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

81 7 06 014

AD A101042

DTIC FILE COPY

AFOSR-TR- 81-0542

Interim Technical Report

for Period 5-1-80 to 4-30-81

Grant [redacted]-AFOSR-80-0158

# VECTORIZED SPARSE ELIMINATION



D. A. Calahan

Principal Investigator

Systems Engineering Laboratory

University of Michigan

Ann Arbor, MI

June 8, 1980

A. INTRODUCTION

The vector processor has reopened for research the study of general sparse matrix algorithms. Such study can be divided into the following classifications.

1. Block-oriented methods that exploit vectors within dense substructures.
2. Simultaneous system methods that exploit global structural symmetries.
3. Fast scalar methods (a new study), for parts of a system for which the above methods cannot be used.

General sparse solvers tend to be used in the following types of applications.

1. *ODE/algebraic systems* such as electrical circuits and rigid body dynamics, where the random nature of the structural regularities requires general sparse solvers. The difficulty here is that vectorization may require major modifications in the equation formulation by the user.
2. Finite element and finite difference systems; here they must compete with solvers written for specific structures such as block triadiagonal and banded systems. With the ability to solve arbitrary structures, they are excellent for vector processor and algorithm evaluation before structurally dedicated assembly coded solvers can be written. It must be acknowledged however, that vector processors favor dense systems and dedicated solvers, so that there is a performance bias against over well-documented and -programmed general solvers.

With these realizations, current research is oriented toward solutions of ODE/algebraic applications. However, opportunities for use of general solvers in solution of finite difference and finite element grids will certainly be exploited.[2]

B. PROGRESS

1. Locally dense systems. A block-oriented general sparse equation solver that exploits cache memory on the CRAY-1 has been prepared and documented [3]; comparisons are now being made between its performance and solvers for specialized common structures. One performance generalization that can be made is that the general solver devotes approximately half of its time to list processing for square blocks of size 3; for larger blocks, the numeric processing kernels predominate.

2. Simultaneous systems. The concept and application of a simultaneous general sparse solver was first ennunciated in [1] for VLSI circuit analysis and in [2] for solution of 2-D finite difference grids. The performance figures given were based on an assembly language (CAL) implementation. A new approach, patterned after the work of Gustavson [4], is now being studied, where explicit, loopless vector code specific to a sparsity pattern is first generated to solve across identically-structured sparse systems. The major objection to Gustavson's original scalar code generation scheme was the code storage required to solve large systems. However, because each vector instruction now performs up to 64 floating point operations in a simultaneous sparse solver, the code storage requirements are potentially (1/64)th that of the equivalent scalar code. A code generation version of our original CAL simultaneous sparse solver is now under development.

3. Fast scalar solvers. Sparse systems inevitably have (hopefully small) components not amenable to vectorized solution. In VLSI circuit analysis, for example, this component corresponds to the interconnection network. Experience indicates that a small scalar computational component can become important if care is not taken in its coding.

Fast scalar equation solvers have been developed to execute on the CRAY-1 in the range of 12-20 MFLOPS. For example, a seminar student at Michigan developed, by careful instruction scheduling with our CRAY-1 simulator, a scalar tridiagonal solver that outperforms vectorized cyclic reduction codes [5]. To replace the interactive instruction scheduling performed by the student, we are examining the use of the CPM for automatically scheduling explicit scalar code generated in the manner of Gustavson [4]. CPM is used for instruction scheduling in the Fortran compilers of both the CRAY-1 and the CYBER 203/205, so that its specialization to sparse matrix codes should not be limited to the CRAY-1 in application. The instructionscheduler should also provide an interesting adjunct to our CRAY-1 cross assembler/simulator software (see section 5 below).

4. VLSI design. A PhD student from UC Berkeley engaged in electronic circuit simulation was a visitor to our research group during the summer of 1980 to study the CRAY-1 and our sparse matrix research. Under subsequent Bell Laboratories auspices, he produced the paper of Appendix A. The sparse solver he developed generates explicit unoptimized scalar code; despite this inefficiency, the sparse matrix solution dropped in significance (visa vis the original, Fortran-coded solver) to 21% of the total analysis time. We have consequently redirected 1-2 months of effort to

assisting him in the vectorization of the now dominant equation formulation (modeling) stage of the circuit analysis. His sparse matrix code has spawned the above-mentioned research on scheduling, however.

5. Simulator development. There is increased interest in our assembler/simulator package* among CRAY-1 users who are dedicated to developing higher performance codes. Because we benefit from interaction with such sophisticated groups, we are devoting 1-2 man-months to remove local operating system dependencies of the software that inhibit conversion to other systems. Also, we are promised early access to serial #1 of the CRAY-2, which is expected to have an instruction set similar to the CRAY-1. The simulator will then be modified accordingly and will become even more useful to us and to others as an evaluation and code development tool.

C. COUPLING ACTIVITIES

A. Seminars

1. On vectorized sparse elimination.
   a. Washington State University, November 17, 1980
   b. University of California, Berkeley, November 19, 1980
   c. University of Texas, Austin, October 30, 1980
2. On other topics in vector processing.
   a. 4-day seminar at AFFDL, June 1980
   b. Seminar at LANL, August 1980

*Mitsubishi (Japan), Daresbury (UK), Exxon, Univ. of Alabama

B. Counsulting

1. Visiting scientist, AFFDL, to give instruction on algorithms for vector processing, and to study I/0 problems associated with Navier-Stokes codes on the CYBER 203/205.
2. Visiting scientist, LANL, on vectorized Monte Carlo.
3. Industrial consultant, Mobil Research and Development, on the vectorization of 3-D diffussion codes associated with oil reservoir drilling and management.

C. Other

A one-week short course at the University of Michigan on High Speed Computation was taught in August, 1980. Among 44 attendees were representatives from NRL and DOD.

## Grant-Supported Publications

### Conference Publications

[1] Calahan, D.A., "Multilevel Vectorized Sparse Solution of LSI Circuits," IEEE International Conference on Circuits and Computers, ICCC 80, October 1-3, 1980, Rye, N.Y., pp. 976-989.

[2] Calahan, D.A., "Sparse Vectorized Direct Solution of Elliptic Problems," Proc. Conf. on Elliptic Problem Solvers, Santa Fe, July 1, 1980. (to be published by Academic Press)

### Reports

[3] Calahan, D.A., "A Block-Oriented Sparse Equation Solver for the CRAY-1," Report #136 , Systems Engineering Laboratory, University of Michigan, December, 1980.

## Other Report References

[4] Gustavson, F.G., W.M. Linger, and R.A. Willoughby, "Symbolic Generation of an Optional Crout Algorithms for Sparse Systems of Linear Equations," Sparse Matrix Proceedings, Conf. at IBM T.J. Watson Res. Cntr., March, 1969.

[5] Brown, F., "A Tridiagonal Solver for the CRAY-1," project report, seminar on High-Speed Computation, University of Michigan, May, 1980.

# APPENDIX A

## Recent Report on VLSI Circuit Analysis

# A COMPUTER PROGRAM FOR THE SIMULATION OF LARGE-SCALE-INTEGRATED CIRCUITS

Andrei Vladimirescu and Donald O. Pederson

Department of Electrical Engineering and Computer Sciences,
Electronics Research Laboratory,
University of California, Berkeley, California 94720

ABSTRACT:

**The algorithms, data structures and semiconductor device modeling techniques proposed for a new simulation program for LSI and VLSI circuits are presented. The selected algorithms and modeling techniques are designed for efficient execution on vector computers.**

## 1. Introduction

The SPICE [1] program has had wide acceptance and experience for integrated circuit evaluation during the last ten years. Although initially designed to simulate efficiently circuits of up to one to two hundred elements, SPICE 2G [2] is presently used to analyze circuits which are one to two orders of magnitude larger. The run-time for a circuit of this extent is measured in days on the VAX 11/780, hours on a CYBER and tens of minutes on the CRAY-1. In spite of an inherent increase in the execution speed of SPICE2 on new main-frame computers, an additional order of magnitude increase in speed is needed for the efficient simulation of LSI circuits.

Several papers have recently been published on algorithms and approaches for the computer-aided circuit analysis of LSI. However only two working circuit simulation programs [3,4] have been reported to date and neither provides the desired speed improvement for LSI circuits.

The different design aspects of the new program are outlined in the following sections. The algorithms and appropriate data structures are described first. The modeling techniques used in timing simulation can be adapted efficiently to the new program as shown next. Comments on the initial results obtained from experiments with SPICE 2G on the CRAY-1 are presented in conclusion.

## 2. Algorithms

Two basic factors of present technology are considered in the design of the new LSI circuit simulator.[1] The first one is that LSI circuits are usually a collection of a limited number of structurally identical functional blocks such as logic gates, operational amplifiers, etc. The second factor is the advent of vector computers which provide the ideal architecture for fast computations on repetitive structures. SPICE2 operates on an entire circuit matrix which is loaded element by element. The analysis in the new program is done at the functional block (subcircuit) level. However, only direct methods are used in the solution procedure, i.e. no decoupling takes place at any level as in timing or mixed-mode simulation [5]. However the structured input description is used to perform an important part of the computation in parallel. Therefore, the new program groups the identical functional blocks together and performs a two-level analysis [6]. Each functional block (subcircuit) generates a diagonal submatrix in the overall circuit matrix. Thus the circuit can be represented as a Bordered Block Diagonal Form (BBDF) matrix. At the lower level there are the diagonal submatrices representing the internal nodes of each subcircuit and at the upper level there is the interconnection matrix (borders and lower-right corner). The linearization and solution of all subcircuits of the same type can be done simultaneously in the vector-mode. This is possible because all these subcircuits have the same topology. The feature of parameter passing between subcircuit call and subcircuit definition increases the occurance of topologically identical subcircuits. This feature provides the possibility of defining a different element value, device or model parameter at each instance of the same subcircuit.

The linear system of equations is solved by executing generated vector code. This approach is used because an important part of the total execution time is spent in equation solution. One set of vector code is generated symbolically for each type of subcircuit. The same code is then used for the LU factorization, forward and backward substitution of all subcircuits of the same type. This approach also reduces the severity of the gather/scatter problem due to provisions taken in the design of the data structure, as described below.

[1] CLASSIE, the acronym for Computer-aided Large-scale Analysis of Solid-State Integrated Electronics (tentative name).

### 3. Data Structure

The dynamic memory management scheme of SPICE2 is preserved for handling the circuit data. However, the data structure which is predominantly of the linked-list type in SPICE2 is changed. The data structure and the main analysis loop are designed to accommodate the new algorithms and to make the vectorization possible. Thus, each element type has its own parameter table and data may be aligned in such a way that vector mode operations can be programmed most effectively. The information on the circuit sparse matrix is organized as one set of sparse submatrix pointers for each type of subcircuit and another set of pointers for the interconnection matrix. The nonzero submatrix entries and RHS for each subcircuit of the same type are aligned and appear as one dataset.

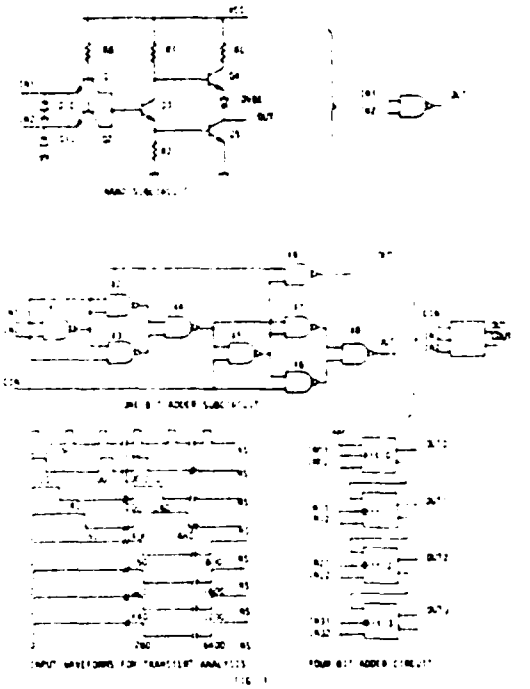
### 4. Device Models

Model evaluation is subject to two constraints. First, it must provide a most accurate representation of devices with continuously changing features due to fast progress in processing. Second, it must be coded in such a way as to allow vectorization by a Fortran compiler. Both requirements can be fulfilled if look-up tables for device data are used instead of mathematical models [7]. The accuracy, speed and ease of implementation are some of the considerations underlying the selection of this modeling approach for circuit simulation [5]. The analytical models available in SPICE 2G can be used as well if recoded for vectorization purposes. Some of the simpler models will be rewritten in order to compare their performance with the table models. Another important aspect of the linearization of semiconductor devices is internal node suppression [8]. Internal nodes are generated due to the existence of series parasitic resistances at the device terminals. Thus for a circuit with 800 nodes and 1000 bipolar transistors if all the parasitic resistances for the emitter, base and collector terminals are specified, the circuit matrix will have an artificial dimension of 3800 rather than the 800 if node suppression is used.

For portability reasons the major part of the new program is coded in Fortran with restrictions required by vector compilers in order to generate vectorized code. The linear equation solution part is coded in assembler language to obtain important speedups of computation.

### 5. Performance Estimate

An estimate of the performance of the new program can be made from various experiments with SPICE 2G on the CRAY-1. A bipolar NAND gate 4-bit adder as shown in Fig. 1 is used as a test circuit (this example can be easily expanded to four or eight times its complexity becoming a 16- or 32-bit adder, respectively). The circuit contains 288 semiconductor devices (180 BJTs and 108 diodes) and is described by a 450*450 matrix. If node suppression is implemented the size of the matrix will be only 270*270 (an additional node is generated by the base resistance of each of the 180 BJTs).



An interesting observation is the time breakdown between model evaluation and equation solution on the CRAY-1. On scalar computers for all-Fortran versions the breakdown between the two is roughly 70%/30% for a 50 equation circuit. The CRAY-1, a pipelined machine, is more efficient in evaluating models rather than tracing through linked lists. Thus, with an all-Fortran version approximately half of the time is spent in the solution part. For a 50 equation circuit 50%, and for a 450 equation circuit 80%, of the total time are spent solving the linearized equations.

An overall factor of two improvement has been obtained just by using generated loopless machine code [9,10] for the equation solution (a factor of 8 for this part only). Some meaningful statistical data on the transient analysis of the circuit in Fig. 1 are summarized in Table 1. The two analyses of Table 1 used five minutes of CRAY-1 cpu time.

TABLE
FORTRAN VS CAL COMPARISON

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| FOR | 1587 | 8648 | 34 | 4% | 84320 | 0 | 1 |
| CAL | 3444 | 18634 | 16 | 17 | 96538 | 2.5 | 2.16 |

The generated code only uses the scalar registers in this case. In the new program there will be identical code templates which use vector registers. The computation speed achieved with the scalar version is 2.5 Mflops. It is estimated the vector version will average 10-20 Mflops at the entire matrix level (subcircuit + interconnection).

## 6. Conclusion

The approach of carefully matching the characteristics of the circuits to be analyzed with the appropriate algorithms and machine architecture make it possible to extend circuit simulation to LSI and VLSI circuits. One order-of-magnitude increase in speed performance is expected from the new program. This brings the program in the same performance category as an existing timing simulator running on the CRAY-1.

## Acknowledgement

The authors gratefully acknowledge the support provided by the Bell Laboratories. The close working relationship with D. A. Calahan has had a significant contribution to the reported work. The stimulating discussions with E. Cohen, A. R. Newton and A. L. Sangiovanni-Vincentelli have proven most useful.

## 7. REFERENCES

[1] L.W.Nagel, "*SPICE2 - A Computer Program to Simulate Semiconductor Circuits*", ERL Memo No. ERL-M520, University of California, Berkeley, May 1975.

[2] A.Vladimirescu, A.R.Newton and D.O.Pederson, "*User's Guide for SPICE2 Version G 1*", University of California, Berkeley, Oct. 1980.

[3] N.G.B.Rabbat, A.L.Sangiovanni-Vincentelli, and H.Y.Hsieh, "A Multilevel Newton Algorithm with Macromodeling and Latency for the Analysis of Large-Scale Nonlinear Circuits in the Time Domain", *Trans IEEE*, Vol. CAS-26, Sept. 1979.

[4] P.Yang, I.N.Hajj, and T.N.Trick, "SLATE: A Circuit Simulation Program with Latency Exploitation and Node Tearing", *Proc.*, Int. Conference on Circuits and Computers, New York, Oct. 1980.

[5] A.R.Newton, "*Timing, Logic and Mixed-Mode Simulation for Large MOS Integrated Circuits*", NATO Advanced Study Institute on Computer Design Aids for VLSI Circuits, Sogesta-Urbino, Italy, July 1980.

[6] D.A.Calahan, "Multi-Level Vectorized Sparse Solution of LSI Circuits", *Proc.*, Int. Conference on Circuits and Computers, New York, Oct. 1980.

[7] B.R.Chawla, H.K.Gummel and P.Kozak, "MOTIS - An MOS Timing Simulator", *Trans. IEEE*, Vol. CAS-22, Dec. 1975.

[8] A.F.Lachner and W.J.Mc. Calla, "Node Suppression in the Simulation of IIL", *Proc.*, IEEE Int. Symposium on Circuits and Systems, N.Y., N.Y., May 1978.

[9] E.Cohen, "Performance Limits of a Dedicated CAD System", *Proc.*, IEEE Int. Symposium on Circuits and Systems, Houston, Texas, April 1980.

[10] E.Cohen, "*Program Reference for SPICE2*", ERL Memo No. ERL-M592, University of California, Berkeley, June 1976.

# DEVICE MODEL VECTORIZATION

| | #TTP | #ITER | CP/IT(MS) | % CP TIME EQN SOL | % CP TIME MODEL | -%Δ T | SPEED |
|---|---|---|---|---|---|---|---|
| ALL FORT | 1507 | 8648 | 34 | 60 | 40 | – | 1 |
| CAL EQN | 3444 | 18034 | 16 | 17 | 78 | 87 | 2.16 |
| VECT.+CAL | 5095 | 27001 | 11 | 21 | 72 | 33 | 3.2 |

180 BJT 108 DIODES 450 EQNS

MAXIMUM VECTOR LENGTH = 64